American Journal of Computational Mathematics, 2011, 1, 147-151
doi:10.4236/ajcm.2011.13016 Published Online September 2011 (http://www.SciRP.org/journal/ajcm)

Computing the Moore-Penrose Inverse of a Matrix through Symmetric Rank-One Updates

Xuzhou Chen[1], Jun Ji[2]
[1]*Department of Computer Science*, *Fitchburg State University*, *Fitchburg*, *USA*
[2]*Department of Mathematics and Statistics*, *Kennesaw State University*, *Kennesaw*, *USA*
E-mail: *xchen@fitchburgstate.edu*, *jji@kennesaw.edu*
Received February 22, 2011; *revised April* 14, 2011; *accepted May* 12, 2011

Abstract

This paper presents a recursive procedure to compute the Moore-Penrose inverse of a matrix A. The method is based on the expression for the Moore-Penrose inverse of rank-one modified matrix. The computational complexity of the method is analyzed and a numerical example is included. A variant of the algorithm with lower computational complexity is also proposed. Both algorithms are tested on randomly generated matrices. Numerical performance confirms our theoretic results.

Keywords: Finite Recursive Algorithm, Moore-Penrose Inverse, Symmetric Rank-One Update

1. Introduction

The computation of the generalized inverse of a matrix has been discussed by numerous papers, from Newton types of iterative schemes to finite algorithms. In particular, the finite recursive algorithms have been investigated by several authors [1-4]. Many of these finite algorithms are based on the computation of the generalized inverse of the rank-one modified matrix.

Our aim is to give a new finite recursive algorithm for computing the Moore-Penrose inverse. The approach is based on the symmetric rank-one updates. The work can be viewed as the generalization of an earlier result on the Moore-Penrose inverse of rank-one modified matrix $A + cd^*$ [5, Theorem 3.1.3]. Numerical tests show that our approach is very effective to the computation of Moore-Penrose inverses for rectangular matrices.

Throughout this paper we shall use the standard notations in [5,6]. C^n and $C^{m\times n}$ stand for the n-dimensional complex vector space and the set of $m\times n$ matrices over complex field C respectively. For a matrix $A\in C^{m\times n}$ we denote $R(A)$, $N(A)$, A^*, and $A^\dagger$ the range, null space, conjugate transpose and Moore-Penrose generalized inverse of A, respectively.

It is well-known that the Moore-Penrose inverse $A^\dagger$ of A can be expressed as $A^\dagger = (A^*A)^\dagger A^*$. Thus, we can write

$$A^\dagger = \left(\sum_{i=1}^{m} r_i r_i^*\right)^\dagger A^* \tag{1}$$

where r_i^* is the i-th row of A. Define

$$A_0 = 0,\ A_l = \sum_{i=1}^{l} r_i r_i^*,\ l = 1,2,\cdots,m, \tag{2}$$

and

$$X_l = A_l^\dagger A^*,\ l = 1,2,\cdots,m. \tag{3}$$

Note that $A_l = A_{l-1} + r_l r_l^*$ is the rank-one modification of A_{l-1}. In the next section we will propose a finite recursive algorithm which effectively computes X_l in terms of X_{l-1} starting from X_0 with the help of the existing rank-one update formulas for the Moore-Penrose inverse. After m iterations, $X_m = A_m^\dagger A^*$ will be finally reached, resulting in the Moore-Penrose inverse $A^\dagger$ of A due to the fact that

$$X_m = A_m^\dagger A^* = \left(\sum_{i=1}^{m} r_i r_i^*\right)^\dagger A^*$$
$$= \left(A^*A\right)^\dagger A^* = A^\dagger$$

Our algorithm will never form $A_l^\dagger$ explicitly and perform any matrix multiplications as stated in (3) at each iteration. Thus it has a low computational complexity. The details of its computational complexity will be included in the paper. To improve its computational complexity, a variant is also proposed. A numerical example is presented in Section 4. We also test and compare the efficiencies of Algorithms 1 and 2 by comparing the CPU times on randomly generated matrices of different sizes.

2. The Finite Recursive Algorithm for Moore-Penrose Inverse

First, let us establish a relation between $\left(A_{l-1}+r_l r_l^*\right)^\dagger$ and $A_{l-1}{}^\dagger$. For each $l=1,2,\cdots,m,$ define

$$\begin{aligned} k_l &= A_{l-1}^\dagger r_l, \\ h_l &= r_l^*\ A_{l-1}^\dagger, \\ u_l &= \left(I - A_{l-1}A_{l-1}^\dagger\right) r_l, \\ v_l &= r_l^*\ \left(I - A_{l-1}^\dagger A_{l-1}\right), \\ \beta_l &= 1 + r_l^*\ A_{l-1}^\dagger r_l. \end{aligned} \tag{4}$$

Theorem 1. *Let A_l ($l=1,2,\cdots,m$) and β_l be defined as in* (2) *and* (4) *respectively. Then $\beta_l \geq 1$ for each l.*

Proof. Obviously, $\beta_1 = 1 + r_1^*\ A_0^\dagger r_1 = 1$. We now only need to prove the result for $2 \leq l \leq m$. Observe that $A_l = \sum_{i=1}^{l} r_i r_i^*$ is positive semi-definite for $1 \leq l \leq m$. Let γ be the rank of A_{l-1}. Then A_{l-1} is unitarily similar to a diagonal matrix, *i.e.*,

$$A_{l-1} = UDU^*$$

where U is a unitary matrix and $D = \text{diag}\ \left(\sigma_1, \sigma_2, \cdots \sigma_\gamma, 0, \cdots, 0\right)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\gamma > 0$. We can write $A_{l-1}{}^\dagger = UD^\dagger U^*$ which is also positive semi-definite due to the fact that $D^\dagger = \text{diag}\left(1/\sigma_1, 1/\sigma_2, \cdots, 1/\sigma_\gamma, 0, \cdots, 0\right)$. Therefore, we have $\beta_l = 1 + r_l^*\ A_{l-1}^\dagger r_l \geq 1$. □

The general result for the Moore-Penrose inverse of a rank-one modified matrix can be found in [5]. For a matrix $A \in C^{m\times n}$, $c \in C^m$, $d \in C^n$, the Moore-Penrose inverse of $A + cd^*$ can be expressed in terms of $A^\dagger$, c, and d with six distinguished cases. Our approach is to apply the result to the sequence (3) where each A_l is positive semi-definite, β_l is real and nonzero in view of Theorem 1 which simplifies the theorem by eliminating three cases. Due to the fact that $(A_{l-1}{}^\dagger)^* = (A_{l-1}{}^*)^\dagger = A_{l-1}{}^\dagger$ we also have $h_l = k_l^*$ and $v_l = u_l^*$ from which two of three other cases can be combined into one. Thus, the six cases of [1] are reduced to only two cases. Moreover, the update formulas can be significantly simplified.

Theorem 2. *The Moore-Penrose inverse of $A_l = A_{l-1} + r_l r_l^*$ defined in* (2) *is as follows.*

1) *If $u_l \neq 0$, then*

$$\begin{aligned} &\left(A_{l-1}+r_l r_l^*\right)\left(A_{l-1}+r_l r_l^*\right)^\dagger \\ &= \left(A_{l-1}+r_l r_l^*\right)^\dagger \left(A_{l-1}+r_l r_l^*\right) \\ &= A_{l-1}A_{l-1}^\dagger + u_l u_l^\dagger \end{aligned} \tag{5}$$

and

$$\left(A_{l-1}+r_l r_l^*\right)^\dagger = A_{l-1}^\dagger - k_l u_l^\dagger - u_l^{*\dagger} k_l^* + \beta_l u_l^{*\dagger} u_l^\dagger \tag{6}$$

2) *If $u_l = 0$, then*

$$\begin{aligned} &\left(A_{l-1}+r_l r_l^*\right)\left(A_{l-1}+r_l r_l^*\right)^\dagger \\ &= \left(A_{l-1}+r_l r_l^*\right)^\dagger \left(A_{l-1}+r_l r_l^*\right) \\ &= A_{l-1}A_{l-1}^\dagger \\ &= A_{l-1}^\dagger A_{l-1} \end{aligned} \tag{7}$$

and

$$\left(A_{l-1}+r_l r_l^*\right)^\dagger = A_{l-1}^\dagger - 1/\beta_l k_l k_l^* \tag{8}$$

Proof. The result follows directly from [5, Theorem 3.1.3] and its proof. Details are omitted. □

Finally, let us turn our attention to establishing the iterative scheme from X_{l-1} to X_l. To this end, we define two auxiliary sequences of vectors $y_{s,t} = A_s{}^\dagger r_t$ and $z_{s,t} = A_s y_{s,t} = A_s\, A_s{}^\dagger r_t$. It is easily seen from

$$A_s^\dagger A_s = \left(A_s^\dagger A_s\right)^* = A_s^*\left(A_s^\dagger\right)^* = A_s\left(A_s^*\right)^\dagger = A_s A_s^\dagger$$

that $y_{s,t}$ and $z_{s,t}$ are the minimum-norm least-squares solutions of the following two auxiliary linear systems respectively

$$A_s y = r_t, \quad A_s z = A_s r_t.$$

In what follows we will frequently employ the fact that $a^\dagger = a^*/\|a\|^2$ for any non-zero column vector a. We will distinguish two cases in our analysis in view of Theorem 2.

Case 1 ($u_l \neq 0$). It is easily seen from Theorem 2 that

$$\begin{aligned} X_l &= \left(A_{l-1}+r_l r_l^*\right)^\dagger A^* \\ &= X_{l-1} - k_l u_l^\dagger A^* - u_l^{*\dagger} k_l^* A^* + \beta_l u_l^{*\dagger} u_l^\dagger A^* \end{aligned} \tag{10}$$

Note that

$$\begin{aligned} k_l u_l^\dagger A^* &= A_{l-1}^\dagger r_l \left[\left(I - A_{l-1}A_{l-1}^\dagger\right) r_l\right]^\dagger A^* \\ &= \frac{A_{l-1}^\dagger r_l r_l^* \left(I - A_{l-1}A_{l-1}^\dagger\right) A^*}{r_l^*\left(r_l - A_{l-1}A_{l-1}^\dagger r_l\right)} \\ &= \frac{y_{l-1,l}\left(r_l^* - \left(A_{l-1}A_{l-1}^\dagger r_l\right)^*\right) A^*}{r_l^*\left(r_l - z_{l-1,l}\right)} \\ &= \frac{y_{l-1,l}\left(r_l - z_{l-1,l}\right)^* A^*}{r_l^*\left(r_l - z_{l-1,l}\right)} \\ &= \frac{y_{l-1,l}\left(A\left(r_l - z_{l-1,l}\right)\right)^*}{r_l^*\left(r_l - z_{l-1,l}\right)} \end{aligned} \tag{11}$$

Similarly, we have

$$u_l^{*\dagger} k_l^* A^* = \frac{\left(r_l - z_{l-1,l}\right)\left(Ay_{l-1,l}\right)^*}{r_l^*\left(r_l - z_{l-1,l}\right)} \tag{12}$$

and

$$\begin{aligned}&\beta_l u_l^{*\dagger} u_l^{\dagger} A^*\\&=\left(1+r_l^* A_{l-1}^{\dagger} r_l\right)\frac{\left(r_l-z_{l-1,l}\right)\left(r_l^*-z_{l-1,l}^*\right)A^*}{\left[r_l^*\left(r_l-z_{l-1,l}\right)\right]^2}\\&=\frac{1+r_l^* A_{l-1}^{\dagger} r_l}{\left[r_l^*\left(r_l-z_{l-1,l}\right)\right]^2}\left(r_l-z_{l-1,l}\right)\left(A\left(r_l-z_{l-1,l}\right)\right)^*\end{aligned} \tag{13}$$

Combining (10) through (13), we have

$$\begin{aligned}X_l &= X_{l-1}-\frac{y_{l-1,l}\left(A\left(r_l-z_{l-1,l}\right)\right)^*}{r_l^*\left(r_l-z_{l-1,l}\right)}-\frac{\left(r_l-z_{l-1,l}\right)\left(Ay_{l-1,l}\right)^*}{r_l^*\left(r_l-z_{l-1,l}\right)}\\&\quad+\frac{1+r_l^* y_{l-1,l}}{\left[r_l^*\left(r_l-z_{l-1,l}\right)\right]^2}\left(r_l-z_{l-1,l}\right)\left(A\left(r_l-z_{l-1,l}\right)\right)^*\end{aligned} \tag{14}$$

It is seen from Theorem 2 that

$$\begin{aligned}y_{l,t} &= A_l^{\dagger} r_t=\left(A_{l-1}+r_l r_l^*\right)^{\dagger} r_t\\&=\left(A_{l-1}^{\dagger}-k_l u_l^{\dagger}-u_l^{*\dagger} k_l^*+\beta_l u_l^{*\dagger} u_l^{\dagger}\right) r_t.\end{aligned}$$

Thus, we have an iterative scheme for $y_{l,t}$:

$$\begin{aligned}y_{l,t} &= y_{l-1,t}-\frac{y_{l-1,l} r_l^*\left(r_t-z_{l-1,t}\right)}{r_l^*\left(r_l-z_{l-1,l}\right)}-\frac{\left(r_l-z_{l-1,l}\right) r_l^* y_{l-1,t}}{r_l^*\left(r_l-z_{l-1,l}\right)}\\&\quad+\frac{\left(1+r_l^* y_{l-1,l}\right)}{\left[r_l^*\left(r_l-z_{l-1,l}\right)\right]^2}\left(r_l-z_{l-1,l}\right) r_l^*\left(r_t-z_{l-1,t}\right)\end{aligned} \tag{15}$$

For the auxiliary sequence $z_{l,t} = A_l y_{l,t}$, we could multiply A_l on both sides of (15) and then simplify the resulted expression. However, in our derivation we employ (5) instead:

$$\begin{aligned}z_{l,t} &= A_l A_l^{\dagger} r_t=\left(A_{l-1}A_{l-1}^{\dagger}+u_l u_l^{\dagger}\right) r_t=z_{l-1,t}+u_l u_l^{\dagger} r_t\\&=z_{l-1,t}+\frac{\left(I-A_{l-1}A_{l-1}^{\dagger}\right) r_l r_l^*\left(I-A_{l-1}A_{l-1}^{\dagger}\right) r_t}{r_l^*\left(I-A_{l-1}A_{l-1}^{\dagger}\right) r_l}\end{aligned}$$

Therefore, we have

$$z_{l,t}=z_{l-1,t}+\frac{\left(r_l-z_{l-1,l}\right) r_l^*\left(r_t-z_{l-1,t}\right)}{r_l^*\left(r_l-z_{l-1,l}\right)}. \tag{16}$$

Case 2. ($u_l = 0$). Observe that

$$\beta_l=1+r_l^* A_{l-1}^{\dagger} r_l=1+r_l^* y_{l-1,l}, \tag{17}$$

$$k_l k_l^* A^*=y_{l-1,l}\left(Ay_{l-1,l}\right)^* \tag{18}$$

It is seen from Theorem 2 that

$$\begin{aligned}X_l &= \left(A_{l-1}+r_l r_l^*\right)^{\dagger} A^*=\left(A_{l-1}^{\dagger}-1/\beta_l k_l k_l^*\right) A^*\\&=X_{l-1}-1/\beta_l k_l k_l^* A^*,\end{aligned}$$

which, together with (17) and (18), implies

$$X_l=X_{l-1}-\frac{1}{1+r_l^* y_{l-1,l}} y_{l-1,l}\left(Ay_{l-1,l}\right)^*. \tag{19}$$

By following the same token, we can develop an iterative scheme for $y_{l,t}$:

$$y_{l,t}=y_{l-1,t}-\frac{r_l^* y_{l-1,t}}{1+r_l^* y_{l-1,l}} y_{l-1,l}. \tag{20}$$

For $z_{l,t}$, in view of (7), we have

$$z_{l,t}=z_{l-1,t}. \tag{21}$$

Finally, since $A_0 = 0$ we have $X_0 = 0$, $y_{0,t} = z_{0,t} = 0$ (t = 1, 2, ···, m) from which we can compute the Moore-Penrose inverse $X_m = A^{\dagger}$ by applying (14) or (19) repeatedly with the help of auxiliary sequences $y_{l,t}$ and $z_{l,t}$ ($t = l + 1$, $l + 2$, ···, m and $l < m$). We summarize this procedure in the following algorithm.

Algorithm 1. MPinverse1[A]

- Step 0 Input: A. Let r_i^* be the i-th row of A (i = 1, 2, ···, m);
- Step 1 Initialization: Set $X_0 = 0 \in C^{n \times m}$, $y_{0,t} = z_{0,t} = 0 \in C^n$ for all t = 1, 2, ···, m;
- Step 2 For l = 1, 2, ···, m,
  1) if $r_l - z_{l-1,l} \neq 0$, then
     compute X_l using (14);
     compute $y_{l,t}$ and $z_{l,t}$ using (15) and (16) respectively for all $t = l + 1, l + 2, \cdots, m$ and $l < m$;
  2) if $r_l - z_{l-1,l} = 0$, then
     compute X_l using (19);
     compute $y_{l,t}$ and $z_{l,t}$ using (20) and (21) respectively for all $t = l + 1, l + 2, \cdots, m$ and $l < m$;
- Step 3 Output: X_m, the Moore-Penrose inverse of A.

Let us analyze the two cases in our algorithm further. If $u_l = 0$ for some l, which is the case Step 2(b) in our algorithm, then we have

$$r_l=A_{l-1}A_{l-1}^{\dagger} r_l=\left(\sum_{i=1}^{l-1} r_i r_i^*\right) A_{l-1}^{\dagger} r_l=\sum_{i=1}^{l-1}\left(r_i^* A_{l-1}^{\dagger} r_l\right) r_i.$$

This means that if $u_l = 0$ for some l, r_l is a linear combination of $\{r_i: I = 1, 2, \cdots, l - 1\}$. The following interesting result shows that the opposite is also true.

Theorem 3. *Let u_i and r_i (i = 1, 2, ···, m) be defined as before. Then, $u_l = 0$ if and only if r_l is a linear combination of $\{r_1, r_2, \cdots, r_{l-1}\}$.*

Proof. The definition of $u_l = (I - A_{l-1}A_{l-1}^{\dagger})r_l$ indicates that $u_l = 0$ is equivalent to $r_l \in N(I - A_{l-1}A_{l-1}^{\dagger})$. Let $\hat{A}_{l-1} = [r_1 | r_2 | \cdots | r_{l-1}]$. Obviously, we have $A_{l-1} = \hat{A}_{l-1}\hat{A}_{l-1}^*$. It is easily seen that

$$R\left(\hat{A}_{l-1}\right) = R\left(\hat{A}_{l-1}\hat{A}_{l-1}^{*}\right) = R\left(A_{l-1}\right)$$
$$= R\left(A_{l-1}A_{l-1}^{\dagger}\right) = N\left(I - A_{l-1}A_{l-1}^{\dagger}\right).$$

Therefore, $u_l = 0$ is equivalent to $r_l \in R(\hat{A}_{l-1})$, *i.e.*, r_l is a linear combination of $\{r_1, r_2, \cdots, r_{l-1}\}$. □

3. A Variant of the Finite Recursive Algorithm with an Improved Computational Complexity

We now compute the computational complexity of Algorithm 1, and then discuss a revised algorithm to improve the efficiencies. Let us focus on the multiplications and divisions ignoring the additions and subtractions. We also count the dominant terms only and ignore the lower terms. Assume that the same quantity is computed only once when the algorithm is implemented. For example, $r_l^{*}(r_l - z_{l-1,l})$ in (14) is computed only once and will not be re-computed in (15)-(16). After optimizing the code, it is not difficult to see that $4mn$ operations are need in (14) and $6n$ in (15)-(16). In Case 2, $2mn$ operations are needed in (19) and $2n$ in (20)-(21). Therefore, the total number of operations T is about

$$T = \sum_{u_l \neq 0}\left(4mn + \sum_{t=l+1}^{m} 6n\right) + \sum_{u_l = 0}\left(2mn + \sum_{t=l+1}^{m} 2n\right)$$
$$= \sum_{u_l \neq 0}\left(4mn + 6n(m-l)\right) + \sum_{u_l = 0}\left(2mn + 2n(m-l)\right).$$

Let γ be the rank of A. In view of Theorem 3, there are exactly γ elements in the set $\{l: 1 \le l \le m, u_l \neq 0\}$ and $(m - \gamma)$ elements in its complement set $\{l: 1 \le l \le m, u_l = 0\}$. Therefore, we have

$$T = 4mn\gamma + \sum_{u_l \neq 0} 6n(m-l) + 2mn(m-\gamma) + \sum_{u_l = 0} 2n(m-l)$$
$$= 2m^2 n + 2mn\gamma + \sum_{u_l \neq 0} 6n(m-l) + \sum_{u_l = 0} 2n(m-l).$$
$$= 2m^2 n + 2mn\gamma + \sum_{u_l \neq 0} 4n(m-l) + \sum_{l=1}^{n} 2n(m-1)$$
$$= 3m^2 n + 2mn\gamma - mn + \sum_{u_l \neq 0} 4n(m-l).$$

Note that

$$\sum_{u_l \neq 0} 4n(m-l) \le 4n\left[(m-l) + (m-2) + \cdots + (m-\gamma)\right]$$
$$= 4n\left(\gamma m - \gamma(\gamma+1)/2\right)$$

Therefore, after ignoring lower terms, we obtain

$$3m^2 n + 2\gamma mn \le T \le 3m^2 n + 6\gamma mn - 2\gamma^2 n, \qquad (22)$$

which leads to the following theorem.

Theorem 4. *When applied to the matrix* $A \in C^{m \times n}$, *Algorithm* 1 *computes the Moore-Penrose inverse of A with roughly T multiplications and divisions where T satisfies* (22).

In view of Theorem 4, the computational complexity of Algorithm 1 grows linearly as a function of n. However, it grows quadratically as a function of m. Thus this algorithm is very fast for small m and much slower for large m. Our preliminary numerical tests confirm the theoretical discovery. Due to the fact that $(A^{*})^{\dagger} = (A^{\dagger})^{*}$, let us apply Algorithm 1 to A^{*} when m is bigger than n. If the output of Algorithm 1 on A^{*} is Y, *i.e.*, Y = MPinverse1$[A^{*}]$, then we have $Y = (A^{*})^{\dagger}$ so $A^{\dagger} = Y^{*}$. We end up this section with the following algorithm for $A^{\dagger}$.

Algorithm 2. MPinversel$[A]$

- Step 0 Input: A;
- Step 1 If $m \le n$, then apply Algorithm 1 to A, *i.e.*, X = MPinverse1$[A]$;
- Step 2 If $m \ge n$, then apply Algorithm 1 to A^{*}. Let Y = MPinverse1$[A^{*}]$ and set $X = Y^{*}$;
- Step 3 Output: X, the Moore-Penrose inverse $A^{\dagger}$ of A.

If Step 2 is called, then the computational complexity of Algorithm 2 is bounded by $3mn^2 + 6\gamma mn - 2\gamma^2 m$ in view of Theorem 4 since the roles of m and n are switched. Therefore, the complexity of Algorithm 2 is bounded by K where $K = 3m^2 n + 6\gamma mn - 2\gamma^2 n$ if $m \le n$ and $K = 3mn^2 + 6\gamma mn - 2\gamma^2 m$ if $m > n$.

4. Preliminary Numerical Results

We show some numerical results obtained by both algorithms proposed in the previous sections. One major advantage of our algorithms is that they will be guaranteed to carry out a result successfully.

We first illustrate our algorithms on a small problem with accurate calculation at each step.

Example 1. Let

$$A = \begin{bmatrix} 1 & 2 & 3 \\ & & \\ 4 & 5 & 6 \end{bmatrix} \qquad (23)$$

By using either one of our algorithms, we generate a sequence of matrices

$$X_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, X_1 = \begin{bmatrix} 1/14 & 8/49 \\ 1/7 & 16/49 \\ 3/14 & 24/49 \end{bmatrix},$$
$$X_2 = \begin{bmatrix} -17/18 & 4/9 \\ -1/9 & 1/9 \\ 13/18 & -2/9 \end{bmatrix} \qquad (24)$$

where $X_2 = A^{\dagger}$ which is the Moore-Penrose inverse of A.

Now, we perform our algorithms on randomly generated matrices using a MATLAB implementation of Al-

gorithms 1 and 2. The MATLAB Profiler is used in the computation of CPU times for the solution of each problem by each algorithm.

Table 1 records the times (in seconds) taken by our algorithm on each randomly generated matrix of various sizes. The recorded results coincide with the computational complexity analysis perfectly. Observe that the performances of both algorithms are exactly the same in *Test* 1 which is expected due to the fact that Algorithm 2 indeed calls Algorithm 1 in this case. However, in the case where $m \gg n$ as in *Tests* 2, 3 and, 4, Algorithm 1 is extremely slow as expected from the computational complexity analysis while Algorithm 2 dramatically reduces the number of operations, thus less time needed for solutions.

5. Conclusions

Two finite recursive algorithms are proposed in this paper for the Moore-Penrose Inverse of a matrix $A \in R^{m\times n}$. They are based on the expression for the Moore-Penrose inverse of symmetric rank-one modified matrix. The computational complexities of both algorithms are analyzed. For matrices with either much less rows than columns or much less columns than rows, the second algorithm is extremely effective according to its computational complexity, which is also confirmed by our preliminary numerical tests on randomly generated matrices of different sizes.

The idea in this paper may be adopted to calculate the minimum-norm least-squares solution to the system of linear equations $A\,x = b$.

Table 1. Computational times (in seconds) for randomly generated matrices.

Algorithm	Test 1	Test 2	Test 3	Test 4
	$m = 20$ $n = 1000$	$m = 2000$ $n = 10$	$m = 2000$ $n = 10$	$m = 10000$ $n = 30$
Algorithm 1	0.063	15.156	74.563	590.985
Algorithm 2	0.063	0.032	0.547	1.235

6. Acknowledgements

The authors are grateful to the referees for their useful comments.

7. References

[1] X. Chen, "The Generalized Inverses of Perturbed Matrices," *International Journal of Computer Mathematics*, Vol. 41, No. 3-4, 1992, pp. 223-236.

[2] T. N. E. Greville, "Some Applications of Pseudoinverse of a Matrix," *SIAM Review*, Vol. 2, No. 1, 1960, pp. 15-22. doi:10.1137/1002004

[3] S. R. Vatsya and C. C. Tai, "Inverse of a Perturbed Matrix," *International Journal of Computer Mathematics*, Vol. 23, No. 2, 1988, pp. 177-184. doi:10.1080/00207168808803616

[4] Y. Wei, "Expression for the Drazin Inverse of a 2 × 2 Block Matrix," *Linear and Multilinear Algebra*, Vol. 45, 1998, pp. 131-146. doi:10.1080/03081089808818583

[5] S. L. Campbell and C. D. Meyer, "Generalized Inverses of Linear Transformations," Pitman, London, 1979.

[6] G. R. Wang, Y. Wei and S. Qiao, "Generalized Inverses: Theory and Computations," Science Press, Beijing/New York, 2004.

